Continual Learning of Knowledge Graph Embeddings

Angel Daruna[1], Mehul Gupta[1], Mohan Sridharan[2] and Sonia Chernova[1]

Abstract— **In recent years, there has been a resurgence in methods that use distributed (neural) representations to represent and reason about semantic knowledge for robotics applications. However, while robots often observe previously unknown concepts, these representations typically assume that all concepts are known a priori, and incorporating new information requires all concepts to be learned afresh. Our work relaxes this limiting assumption of existing representations and tackles the incremental knowledge graph embedding problem by leveraging the principles of a range of continual learning methods. Through an experimental evaluation with several knowledge graphs and embedding representations, we provide insights about trade-offs for practitioners to match a semantics-driven robotics applications to a suitable continual knowledge graph embedding method.**

I. Introduction

Representing and reasoning about semantic knowledge is a key task in robotics. In recent years, there has been a resurgence in methods that use distributed (neural) representations, e.g., word and knowledge graph embeddings, for this task in the context of navigation [1], grounding [2], affordance modeling [3], success detection [4], manipulation [5], and instruction following [6]. While robots frequently observe previously unknown concepts, these embedding algorithms typically assume that all embedding concepts are known a priori, and incorporating new information requires all concepts to be learned afresh. In addition, in robotics applications, the limited availability of computational resources and storage, and concerns regarding storing sensitive information, can make batch learning with all observed data infeasible. We seek to relax this static assumption in knowledge graph embedding and enable adaptive revision of distributed representation of semantic knowledge for robots.

Towards achieving our objective, we draw on *Continual Learning*, the research area which focuses on the challenging problem of incrementally revising learned neural representations [7]. Existing continual learning methods have predominantly been applied to object recognition and include regularization [8], [9], architecture modification [10], [11], generative replay [12], [13], and a reformulation of regularization for knowledge graph embedding [14]. However, continual learning methods remain largely unexplored for knowledge graph embedding. Furthermore, the implications of any related assumptions for robotics is not well documented because existing methods focus on the final inference performance and define different task specific measures [15].

Our work makes three contributions. First, we reformulate and extend the underlying principles of five representative continual learning methods: (i) Progressive Neural Networks [10]; (ii) Copy Weight Re-Init [11]; (iii) L2 regularization [9]; (iv) Synaptic Intelligence [16]; and (v) Deep Generative Replay [12], and apply them to the *continual knowledge graph embedding* (CKGE) problem. Second, we introduce an empirically evaluated heuristic sampling strategy to generate CKGE datasets from knowledge graphs, since benchmark datasets do not exist for the CKGE problem. Third, we build on existing continual learning measures [17] to characterize the use of each reformulated method for robot tasks that leverage semantic knowledge.

For evaluation, we consider two knowledge graph embedding representations with different assumptions and loss functions: TransE [18] and Analogy [19]; and three benchmark knowledge graphs (WN18RR, FB15K237 [20], and AI2Thor [3]). We also evaluated each adapted method under unconstrained, data-constrained, and time-constrained settings by sampling from a knowledge graph used in prior robotics work [3], containing actions, locations, objects, and other concepts. Experimental results indicate that: (i) our generative replay approach outperforms other methods; (ii) there are interesting trade-offs between inference capability, learning speed, and memory usage that should be considered when choosing a CKGE method; and (iii) insights gained from exploring these trade-offs enable us to select a CKGE method that best matches the constraints of a given robotics application that models semantic knowledge.

II. Related Work & Background

We motivate our contributions by discussing background information and related work.

Modeling Semantic Knowledge in Robotics is often achieved using an explicit model of world semantics in the form of a knowledge graph $\mathcal{G}$ composed of individual facts or triples (h, r, t); h and t are the head and tail entities (respectively) for which the relation r holds, e.g., (*cup, hasAction, fill*) [21], [22], [23], [24]. Recent work has modeled $\mathcal{G}$ using distributed representations because of their ability to approximate proximity of meaning from vector computations [1], [2], [3], [4], [5], [6].

Multi-relational (knowledge graph) embeddings are distributed representations that model $\mathcal{G}$ in vector space [25],

[1] Georgia Institute of Technology, Atlanta, GA. Email: {adaruna3, mgupta320, chernova}@gatech.edu

[2] University of Birmingham, Birmingham, UK. Email: m.sridharan@bham.ac.uk

This work is supported in part by NSF IIS 1564080, NSF GRFP DGE-1650044, and ONR N00014-16-1-2835. Mohan Sridharan was supported in part by ONR N00014-17-1-2434. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the supporters.

learning a continuous vector representation from a dataset of triples $\mathcal{D}=\big\{(h,r,t)_i, y_i \,|\, h_i,t_i\in\mathcal{E}, r_i\in\mathcal{R}, y_i\in\{0,1\}\big\}$, with $i\in\{1...|\mathcal{D}|\}$. Here y_i denotes whether relation $r_i \in \mathcal{R}$ holds between $h_i, t_i \in \mathcal{E}$. Each entity $e\in\mathcal{E}$ is encoded as a vector $\mathbf{v}_e\in\mathbb{R}^{d_\mathcal{E}}$, and each relation $r\in\mathcal{R}$ is encoded as a mapping between vectors $\mathbf{W}_r \in \mathbb{R}^{d_\mathcal{R}}$, where $d_\mathcal{E}$ and $d_\mathcal{R}$ are the dimensions of vectors and mappings respectively [25], [26]. The embeddings for $\mathcal{E}$ and $\mathcal{R}$ are typically learned using a scoring function $f(h,r,t)$ that assigns higher (lower) values to positive (negative) triples [26]. The learning objective is thus to find a set of embeddings $\Theta = \big\{\{\mathbf{v}_e | \, e \in \mathcal{E}\}, \{\mathbf{W}_r | \, r \in \mathcal{R}\}\big\}$ that minimizes the loss $\mathcal{L}_\mathcal{D}$ over $\mathcal{D}$. Loss $\mathcal{L}_\mathcal{D}$ can take many forms depending on the multi-relational embedding representation used, e.g., Margin-Ranking Loss [18] or Negative Log-Likelihood Loss [19]. However, all entities and relations are assumed to be known before training [25], [26], which may be infeasible for robots observing new concepts or new facts about existing concepts.

Continual learning has evolved as a subarea of life-long machine learning. It focuses on neural networks and seeks to learn new domains, classes, or tasks over time without forgetting previously learned knowledge [7]. Methods proposed in the narrow context of object recognition include regularization [8], [9], architecture modification [10], [11], and generative replay [12], [13]. We explore and adapt five representative methods [9], [10], [11], [12], [16]. Different categories of continual learning scenarios exist in the literature based on whether there are shifts in the input or output distributions, and whether the inputs and outputs share the same representation space [27]. Among existing categories, we chose *Incremental Class Learning* (ICL) because it best matches the assumptions of robot systems representing semantic knowledge, with the distribution of input data and target labels changing across learning sessions as the robot incrementally observes disjoint sets of new facts about new and existing concepts.

In CKGE, the dataset $\mathcal{D}$ of a knowledge graph $\mathcal{G}$ is split into multiple datasets $\mathcal{D}^n$ where n indicates the learning session [14]. Each $\mathcal{D}^n$ contains a disjoint set of all triples of a subset of entities and relations. For a robot observing new facts, the size of the set of observed entities, relations, and triples grows, (e.g., $|\mathcal{E}^n| \leq |\mathcal{E}^{n+1}|$), and the embedding must consider new facts and concepts in each learning session. In such a learning scenario, the objective is to find a set of embeddings $\Theta^n = \big\{\{\mathbf{v}^n_e | \, e \in \mathcal{E}^n\}, \{\mathbf{W}^n_r | \, r \in \mathcal{R}^n\}\big\}$ that minimize the loss $\mathcal{L}_{\mathcal{D}^n}$ over the dataset for all time steps. Of the range of continual learning methods, only L2-regularization has been applied to CKGE [14]; more sophisticated methods that have shown promise in other domains, e.g., generative replay, remain unexplored. Also, important measures for robotics, such as learning efficiency and model complexity, are not well documented for representative techniques [17], making it difficult to evaluate the suitability of these methods for modeling semantic knowledge in robotics. Our work is designed to fill these gaps.

Dynamic Graph Embedding is a related approach focused on modeling dynamic graphs with applications to social networks, biology, computational finance, and other domains [28]. A work from [28] on dynamic *knowledge* graph embedding [29] assumes the knowledge graph in each time step is complete, and models the changes in the knowledge graph across learning sessions taking as inputs the knowledge graph from the prior and current learning session. Our work uses a different set of assumptions because we model a scenario where a robot is observing disjoint subsets of a complete knowledge graph. We view our approach as a variant of dynamic knowledge graph embedding in which only a subset of the complete knowledge graph is available for training each learning session. As a result, our algorithms need to take into account problems of catastrophic forgetting. Catastrophic forgetting [7] occurs when a neural representation that was optimized for a prior dataset is trained with new dataset. The neural network's weights are tuned to the new dataset, resulting in a potential loss in performance for classes and tasks not included in the new dataset.

III. Continual Knowledge Graph Embedding

We seek to characterize the use of continual learning methods for knowledge graph embedding in robotics by exploring the associated assumptions and trade-offs. In this section, we describe how we reformulate and extend the principles of five carefully selected representative continual learning techniques to develop *continual knowledge graph embedding* (CKGE) methods. These methods were designed for traditional neural networks and required varying levels of innovation to support knowledge graph embeddings. In each case, we carefully considered the suitability of its principles to support the desired capabilities and assumptions of knowledge graph embeddings.

A. Architectural Modification Methods

Among the methods that modify the architecture of a neural network to accommodate new training data while minimizing performance losses over older data, we adapted two methods for knowledge graph embeddings.

Progressive Neural Networks (PNN) [10] add copies of existing layers of a multi-layered neural network for each new learning session. When a new learning session begins, existing weights are frozen so that back-propagated gradients do not affect the performance over data from previous sessions. Also, lateral connections are made between successive layer copies to enable the forward transfer of previously learned weights. To make PNNs applicable to knowledge graph embedding, we first expand the embedding matrices $\mathbf{v}^n\in\mathbb{R}^{|\mathcal{E}^n|\times d_\mathcal{E}}$ and $\mathbf{W}^n\in\mathbb{R}^{|\mathcal{R}^n|\times d_\mathcal{R}}$ to include new entities and relations in the learning session n. Second, we freeze embeddings for entities and relations encountered in prior learning sessions to prevent their corruption in the current learning session. Instead of creating separate copies of these embedding matrices for each learning session, we only expand the existing matrices to promote forward transfer of prior embeddings in new learning sessions.

Copy Weight with ReInit (CWR) [11] maintains the weights of the final layer of the network during a new learning session (i.e. temporary weights, TW), separate from the corresponding weights trained in prior learning sessions (i.e. consolidated weights, CW) to avoid corruption. Other than the two sets of final layer weights considered during (continual) learning, the weights of other layers are frozen and shared across learning sessions. TW are re-sized and re-initialized in each learning session according to the number of classes being trained. After each learning session, the TW for new classes are copied over to CW, which acts as a memory buffer separate from the network. If a previously trained class is encountered, relevant entries in TW are averaged with those in CW. Training for the subsequent session begins by re-sizing and re-initializing TW.

To apply the principles of CWR to knowledge graph embedding, we first introduce two sets of embeddings: consolidated embeddings (CE) $\{\mathbf{v}_{ce}^n, \mathbf{W}_{ce}^n\}$ and temporary embeddings (TE) $\{\mathbf{v}_{te}^n, \mathbf{W}_{te}^n\}$. Second, for each learning session, we resize and re-initialize the TE for entities $\mathbf{v}_{te}^n$ and relations $\mathbf{W}_{te}^n$ based on the number of entities and relations (respectively) in the session. After the session, we move TE into CE by copying new embeddings or averaging existing ones. As a result, the number of CE increases monotonically in each learning session with the number of observed entities $\mathcal{E}^n$ and relations $\mathcal{R}^n$ so that $\mathbf{v}_{ce}^n \in \mathbb{R}^{|\mathcal{E}^n| \times d_\mathcal{E}}$ and $\mathbf{W}_{ce}^n \in \mathbb{R}^{|\mathcal{R}^n| \times d_\mathcal{R}}$) (respectively); the number of TE changes in each learning session according to the number of entities and relations in that learning session's dataset $\mathcal{D}^n$.

B. Regularization Methods

Freezing previously learned weights prevents their corruption in subsequent sessions, but also prevents shared weights from being revised to better accommodate new concepts. Some continual learning methods allow adjustments to shared weights that perform well for prior and new sessions; they do so by enforcing some regularization terms in new learning sessions. We reformulate two such approaches for knowledge graph embeddings.

L2 Regularization (L2R) [9], [14], [27] is adapted in our approach by adding a regularization term to the learning session loss $\mathcal{L}_{\mathcal{D}^n}$, encouraging the trained weights to not deviate from their previous values:

$$\mathcal{L}_{\mathcal{D}^n} + \lambda \cdot \Big(||\mathbf{v}_e^n - \mathbf{v}^{n-1}||_2^2 + ||\mathbf{W}_r^n - \mathbf{W}^{n-1}||_2^2 \Big) \tag{1}$$

where $e \in \mathcal{E}^{n-1}$, $r \in \mathcal{R}^{n-1}$, and λ is a regularization scaling term tuned as a hyper-parameter. L2R can be rather strict because it penalizes all dimensions of an embedding equally, whereas a subset of the embedding dimensions often contribute more to loss or predictive abilities than others.

Synaptic Intelligence (SI) [16] extends L2R by considering the weight-specific contributions to the reduction in loss over a learning session. These contributions are quantified by summing the gradients that each weight adjustment contributes to the loss and using the total loss reduction as a normalizer. SI is generic enough to apply to knowledge graph embeddings with minimal changes because it is formulated in terms of the weight and loss trajectories. Equation 2 defines our implementation of SI for knowledge graph embedding, re-using terms from [16]:

$$\mathcal{L}_{\mathcal{D}^n} + \lambda \cdot \Big(||\Omega_e(\mathbf{v}_e^n - \mathbf{v}^{n-1})||_2^2 + ||\Omega_r(\mathbf{W}_r^n - \mathbf{W}^{n-1})||_2^2 \Big) \tag{2}$$

where $e \in \mathcal{E}^{n-1}$, $r \in \mathcal{R}^{n-1}$, Ω is the parameter regularization strength [16], and λ is a regularization scaling term tuned as a hyper-parameter for a particular representation. Elastic Weight Consolidation (EWC) [9] was also considered but not used because the assumptions made by the Fisher Information matrix of EWC are not satisfied by many knowledge graph embeddings, e.g., those using Margin-Ranking Loss.

C. Generative Replay Methods

Instead of maintaining model weights across learning sessions, generative replay methods learn generative models of the distribution of training data from previous learning sessions. Then, batch learning is approximated by sampling from the learned distribution and the training data from the current learning session. We reformulate one such method for knowledge graph embeddings.

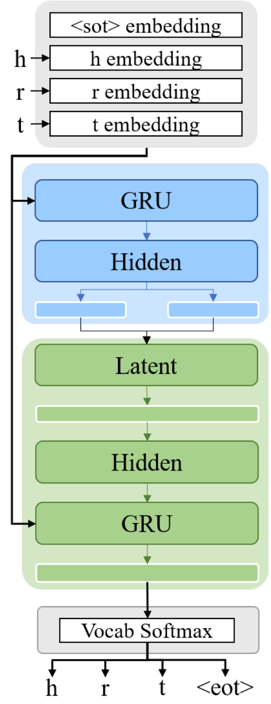


Fig. 1: DGR architecture. Layers with white outline are linear.

Deep Generative Replay (DGR) [12], [13] is a continual learning method that uses a generative model $\mathbf{G}$ to approximate the distribution of all observed training examples (i.e. $\mathcal{D}$), and trains a discriminative model (i.e., solver) to perform a task. In the initial learning session, generator $\mathbf{G}^0$ and solver are trained using examples in $\mathcal{D}^0$. In any subsequent learning session i, a new generator $\mathbf{G}^i$ and solver are trained using examples in $\mathcal{D}^i$ and samples from $\mathbf{G}^{i-1}$ that approximate $\mathcal{D}^{i-1}$, thus approximating training with $\mathcal{D}^{i-1} \cup \mathcal{D}^i$.

The challenge in applying the principles of DGR to knowledge graph embeddings is designing an effective generator, as the solver is determined by the representation used, i.e., $\Theta = \big\{\{\mathbf{v}_e | e \in \mathcal{E}\}, \{\mathbf{W}_r | r \in \mathcal{R}\}\big\}$). Sampling training examples is a known problem in knowledge graph embedding[1], but prior work has shown that a Variational Auto-Encoder (VAE) can be used to sample sequences of discrete tokens [33]. We treat each triple as a sequence of discrete tokens to design our VAE-based generator.

Figure 1 shows our VAE architecture that uses Gated-Recurrent Units to encode and decode the triples to and from the latent space z. Input triples (h, r, t) to the encoder are first transformed into token embedding sequences $x =$

[1] Others have used GANs to generate negative examples [30], [31], [32], but we cannot use these methods because their generators require positive examples as input.

(ν_h, ν_r, ν_t), where $\nu \in \mathbb{R}^{|\mathcal{E}^n|+|\mathcal{R}^n| \times d_\mathcal{V}}$ is a token embedding learned by the encoder with dimensionality $d_\mathcal{V}$. The encoder, shown in blue in Figure 1, is a learned posterior recognition model $q(z|x)$ that approximates the posterior distribution over z, conditioned on the input triple sequences x. Unlike a standard auto-encoder, the encoder is encouraged to keep the learned posterior $q(z|x)$ close to the prior over the latent space $p(z)$, which is a standard Gaussian. A similarity constraint based on the KL divergence measure in the objective function allows samples to be generated from the latent space. These samples are decoded using the decoder, shown in green in Figure 1, to maximize $p(x|z)$, the likelihood of a triple sequence x conditioned on its encoded latent space vector z, as in a standard auto-encoder. The output sequences of the VAE are transformed back into a triples using a Softmax function over all tokens (i.e., $e \in \mathcal{E}^n$ and $r \in \mathcal{R}^n$). The objective function for this architecture is:

$$-\text{KL}\big(q(z|x)||p(z)\big) \cdot \alpha(\text{epoch}) + \mathbb{E}_{q(z|x)}\big[\log p(x|z)\big] \tag{3}$$

where an additional term $\alpha(\cdot)$ is included to anneal the KL divergence loss, preventing issues such as vanishing gradients caused by posterior sampling and KL divergence loss terms being driven to zero [33]. $\alpha(\cdot)$ is a function of the number of epochs trained for the learning session:

$$\alpha(\text{epoch}) = \frac{\lambda_{am}}{1 + e^{-\lambda_{as}\big(\text{epoch} - \lambda_{ap}\big)}} \tag{4}$$

where λ_{am}, λ_{as}, and λ_{ap} are hyper-parameters tuned during training to control the maximum value, slope, and position of the annealing function, respectively.

IV. Experimental Setup

We evaluate our CKGE methods on two multi-relational embedding representations: TransE [18] and Analogy [19]; and three benchmark knowledge graphs: AI2Thor [3], FB15K237 [20], and WN18RR [20]. The last two knowledge graphs are challenging and have been widely used in the graph embedding literature [20], [31], [34]. AI2Thor contains relations and entities related to service robotics, e.g., locations of objects, actions that can be performed on objects, and the outcomes that result from these actions [3]. We report the accuracy and complexity of each method based on seven performance measures chosen from prior continual learning work in robotics [17]. In each trial, the evaluation task is triplet prediction, a fundamental knowledge graph embedding task [1], [3] with a well-defined experimental setup [18], [25] as described later in this section.

TABLE I: CKGE Datasets; Benchmarks

	WN18RR-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	20,368/(50%)	20,389/(73%)	20,249/(87%)	20,463/(95%)	20,437/(99%)
$\|R^n\|$	11/(100%)	11/(100%)	11/(100%)	11/(100%)	11/(100%)
$\|\mathcal{D}^n_{Tr}\|$	17,367/(20%)	17,367/(40%)	17,367/(60%)	17,367/(80%)	17,367/(100%)
$\|\mathcal{D}^n_{Va}\|$	1,117/(37%)	1,141/(57%)	1,187/(71%)	1,190/(80%)	1,184/(86%)
$\|\mathcal{D}^n_{Te}\|$	1,168/(37%)	1,159/(57%)	1,218/(72%)	1,173/(81%)	1,175/(87%)
	FB15K237-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	13,143/(90%)	13,106/(96%)	13,115/(98%)	13,089/(99%)	13,163/(100%)
$\|R^n\|$	237/(100%)	237/(100%)	237/(100%)	237/(100%)	237/(100%)
$\|\mathcal{D}^n_{Tr}\|$	54,423/(20%)	54,423/(40%)	54,423/(60%)	54,423/(80%)	54,423/(100%)
$\|\mathcal{D}^n_{Va}\|$	17,013/(97%)	16,929/(99%)	16,917/(100%)	16,882/(100%)	16,905/(100%)
$\|\mathcal{D}^n_{Te}\|$	19,776/(97%)	19,727/(99%)	19,734/(99%)	19,758/(100%)	19,801/(100%)

TABLE II: CKGE Datasets; Robotics

	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	176/(84%)	175/(95%)	177/(97%)	171/(99%)	169/(99%)
$\|R^n\|$	11/(100%)	11/(100%)	11/(100%)	11/(100%)	11/(100%)
$\|\mathcal{D}^n_{Tr}\|$	17,367/(20%)	17,367/(40%)	17,367/(60%)	17,367/(80%)	17,367/(100%)
$\|\mathcal{D}^n_{Va}\|$	1,117/(37%)	1,141/(57%)	1,187/(71%)	1,190/(80%)	1,184/(86%)
$\|\mathcal{D}^n_{Te}\|$	1,168/(37%)	1,159/(57%)	1,218/(72%)	1,173/(81%)	1,175/(87%)

CKGE datasets: Since there is no established benchmark dataset for CKGE, we introduce three standard evaluation datasets that we obtain by sampling. Our heuristic sampling strategy emulates the New Instances and Concepts scenario presented in [17] under the categorization of the nature of data samples within training sets. Therefore, our sampling strategy models the scenario where a robot explores a world and discovers new triple instances that contain new concepts (i.e. entities or relations), new triple instances that contain previously observed concepts, and triple instances that have been previously observed. Consider a knowledge graph $\mathcal{G}$ whose triples $\mathcal{D}$ have been split into a training set $\mathcal{D}_{Tr}$, validation set $\mathcal{D}_{Va}$, and test set $\mathcal{D}_{Te}$. Our approach for generating datasets for $n = \{1, ..., N\}$ learning sessions is:

1. *Sample training triples*: uniformly sample without replacement $\frac{|\mathcal{D}_{Tr}|}{N}$ triples from training set $\mathcal{D}_{Tr}$ of $\mathcal{G}$. These triples form training dataset $\mathcal{D}^n_{Tr}$.
2. *Extract entities and relations*: create a set of entities E^n and a set of relations R^n for this session from the triples in $\mathcal{D}^n_{Tr}$. The set of all observed entities (relations), i.e., $\mathcal{E}^n$ ($\mathcal{R}^n$) is the union of current and prior E^n (R^n).
3. *Construct n^{th} validation and test sets*: extract from $\mathcal{D}_{Va}$ and $\mathcal{D}_{Te}$ the triples whose head, relation, and tail belong to E^n and R^n (respectively). These triples form validation set $\mathcal{D}^n_{Va}$ and test set $\mathcal{D}^n_{Te}$ of the n^{th} session.
4. *Remove sampled training triples*: remove $\mathcal{D}^n_{Tr}$ from $\mathcal{D}_{Tr}$ of $\mathcal{G}$.
5. Repeat steps 1-4 until no training triples exist in $\mathcal{G}$ or a predefined number of iterations are completed.

We generated three CKGE datasets with $n = 5$ sessions using our approach on two established benchmark knowledge graphs in the graph embedding community (WN18RR and FB15K237 [20]) and a knowledge graph used in robotics (AI2Thor [3]). Tables I and II report statistics of each dataset. The columns of the tables denote the learning session (LS-X, X$\in$ $[1, 5]$), while rows correspond to the statistics, e.g., $|E^n|$ is the size of the entity set. Individual cells indicate the value, with coverage with respect to the original knowledge graph shown in parentheses. For instance, in LS-2 of WN18RR-5-LS, there are $20,389$ entities and 73% of all entities in WN18RR have been observed. Note that our sampling strategy empirically produces datasets with better coverage and higher percentages of new training triples each learning session, i.e., more challenging datasets for CKGE, than previous methods such as entity sampling [14]. Furthermore, our sampling strategy makes the distribution of the n training sets more closely match the original $\mathcal{D}_{Tr}$ than entity sampling by ensuring sampling without replacement $\big(\mathcal{D}^n_{Tr} \bigcap \mathcal{D}^{n+1}_{Tr} = \emptyset \,\forall\, n\big)$.

Evaluation procedure: The evaluation task is to predict complete triplets from incomplete ones in test splits $\mathcal{D}^n_{Te}$, i.e., predict h given (r, t) or t given (h, r). To perform triplet prediction, each test triplet (h, r, t) is first corrupted by replacing the head (or tail) entity with every other possible entity in the current session $\mathcal{E}^n$. Then, to avoid underestimating the embedding performance, we remove all corrupted test triplets that still represent a valid relationship between the corresponding entities; this is known as the "filtered" setting in the literature [18]. Last, scores are computed for each test triplet and its (remaining) corrupted triplets using the scoring function $f(h, r, t)$ (defined below), then ranked in descending order.

Recall that we consider two knowledge graph embedding representations to show the generality of our methods: TransE and Analogy. TransE represents relationships as translations between entities, i.e., $\mathbf{v}_h + \mathbf{W}_r = \mathbf{v}_t$ [18]. It uses the scoring and margin ranking loss functions in Equations 5 and 6, where $[x]_+ = max(0, x)$, γ is the margin, and (h', r, t') are corrupted triples in a corrupted knowledge graph $\mathcal{G}'$. Embeddings are subject to normalization constraints (i.e. $||\mathbf{v}_e||_2 \leq 1 \,\forall\, e \in \mathcal{E}$ and $||\mathbf{W}_r||_2 \leq 1 \,\forall\, r \in \mathcal{R}$) to prevent trivial minimization of $\mathcal{L}$ by increasing entity embedding norms during training.

$$f(h, r, t) = ||\mathbf{v}_h + \mathbf{W}_r - \mathbf{v}_t||_1 \tag{5}$$

$$\mathcal{L} = \sum_{\substack{(h,r,t)\in\mathcal{G},\\ (h',r,t')\in\mathcal{G}'}} [f(h, r, t) + \gamma - f(h', r, t')]_+ \tag{6}$$

Analogy, on the other hand, represents relationships as (bi)linear mappings between entities, i.e., $\mathbf{v}_h^\top \mathbf{W}_r = \mathbf{v}_t^\top$ [19]. It uses the scoring and negative log loss functions in Equations 7 and 8 where σ is a sigmoid function, y is a label indicating whether the triple is corrupted, and $\mathcal{G}'$ is the corrupted knowledge graph. Additionally, the linear mappings (i.e. relations) are constrained to form a commuting family of normal mappings, i.e., $\mathbf{W}_r\mathbf{W}_r^\top = \mathbf{W}_r^\top\mathbf{W}_r \,\forall\, r \in \mathcal{R}$ and $\mathbf{W}_r\mathbf{W}_{r'} = \mathbf{W}_{r'}\mathbf{W}_r \,\forall\, r, r' \in \mathcal{R}$, to promote analogical structure within the embedding space.

$$f(h, r, t) = \langle \mathbf{v}_h^\top \mathbf{W}_r, \mathbf{v}_t \rangle \tag{7}$$

$$\mathcal{L} = \sum_{(h,r,t,y)\in\,\mathcal{G},\mathcal{G}'} -\log\sigma(y \cdot f(h, r, t)) \tag{8}$$

Evaluation measures: We build on existing measures to characterize each of our CKGE methods. We consider different factors important for robotics applications modeling semantic knowledge, e.g., inference, memory usage, and learning efficiency. In addition to the only measure provided in prior CKGE work [14] (i.e. inference performance), we report seven other robotics-oriented metrics cataloged in [17] that measure unique aspects of continual learning algorithms. Specifically, for inference performance, we consider the *mean reciprocal rank* of correct triplets (MRR) and the proportion of the correct triplets ranked in the top 10 (Hits@10). During each learning session, we compute the evaluation measures for the test sets of all learning sessions to characterize the effect of learning on prior, current, and future learning sessions. During the n^{th} learning session of N total sessions, the two training-test inference performance matrices $\mathbf{M} \in \mathbb{R}^{N\times N}$ (for MRR and Hits@10) are used to compute four measures that summarize accuracy and forgetting across learning sessions: (i) Average accuracy (ACC) measures the average accuracy across learning sessions—Equation 9; (ii) Forward Transfer (FWT) measures zero-shot learning in future sessions by transferring weights learned in prior session(s)—Equation 10; (iii) Backwards Transfer (+BWT) measures the improvement over expected performance of a prior learning session as a result of learning in future sessions—Equation 12; and (iv) Remembering (REM) measures how performance in a learning session degrades as a result of learning in subsequent sessions—Equation 13.

$$\text{ACC} = \frac{\sum_{i\geq j}^{N} \mathbf{M}_{i,j}}{\frac{N(N+1)}{2}} \quad (9) \qquad \text{FWT} = \frac{\sum_{i<j}^{N} \mathbf{M}_{i,j}}{\frac{N(N-1)}{2}} \quad (10)$$

$$\text{BWT} = \frac{\sum_{i=2}^{N}\sum_{j=1}^{i-1}(\mathbf{M}_{i,j} - \mathbf{M}_{j,j})}{\frac{N(N-1)}{2}} \tag{11}$$

$$\text{+BWT} = \max(0, \text{BWT}) \quad (12) \qquad \text{REM} = 1 - |\min(0, \text{BWT})| \quad (13)$$

Other measures important for robotics applications that leverage semantics are space complexity and learning speed [17]. We capture space complexity for each CKGE method using Model Size (MS) and Samples Storage Size (SSS) measures [17]. MS measures the growth in memory usage $\mathcal{U}$ for model parameters θ across learning sessions for a particular method—Equation 14. Samples Storage Size (SSS) measures the growth in memory usage $\mathcal{U}$ for stored samples SS across learning sessions as a proportion of the total number of training samples for the task, i.e., $\mathcal{D}_{Tr}$, in Equation 15. For learning speed, we use the Learning Curve Area (LCA) measure [17], which we modify to range between zero and one (like other measures). For a performance measure m, it computes the area covered by the learning curve of the learning method up to the best measured performance m^* at time t as a proportion of the area achieved by perfect zero-shot learning (Equation 16).

$$\text{MS} = \min(1, \frac{\sum_{.E}^{N} \frac{\mathcal{U}(\theta_1)}{\mathcal{U}(\theta_i)}}{N}) \tag{14}$$

$$\text{SSS} = 1 - \min(1, \frac{\sum_{i=1}^{N} \frac{\mathcal{U}(SS_i)}{\mathcal{U}(\mathcal{D}_{Tr})}}{N}) \tag{15}$$

$$\text{LCA} = \frac{\int_0^t m\, dm}{m^* \times t} \tag{16}$$

Software implementation: Please see supplementary material[2] for details about the tuning of hyper-parameters of CKGE methods, each knowledge graph embedding representation used for evaluation, and evaluation datasets, experiments, and results that are omitted here for brevity.

V. Experimental Results

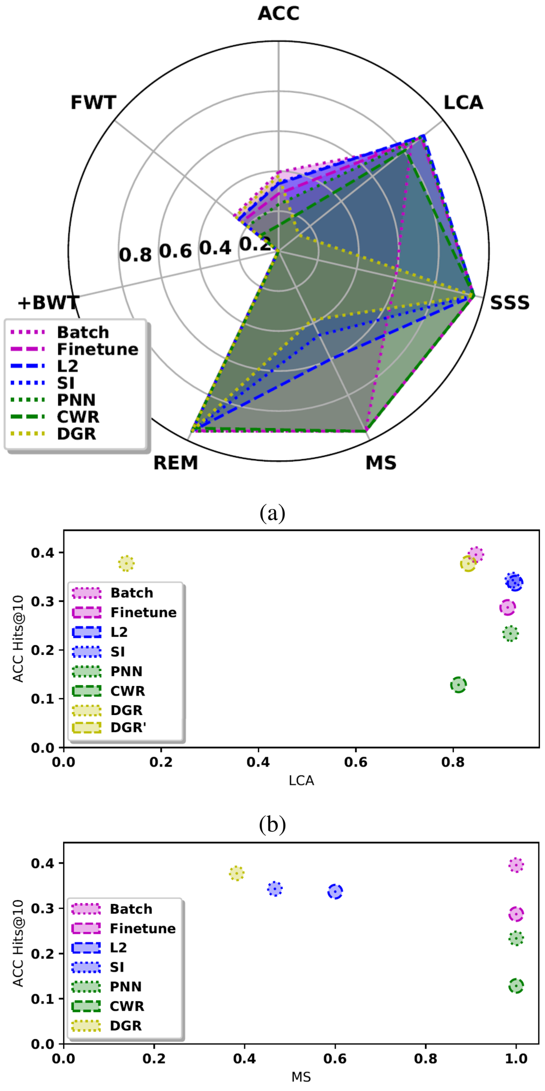
Results reported in this section are the average of five test runs in each experimental scenario; statistical significance is tested using repeated-measures ANOVA and a post-hoc Tukey's test. Any mention of 'significance' implies statistical significance at 95% significance level (i.e. $p < 0.05$).

[2] https://github.com/adaruna3/continual-kge

In addition to the CKGE methods, we considered two additional methods that served as upper and lower bounds (i.e., baselines) for the expected inference performance of the CKGE methods. *Batch* represents the inference upper bound because it can store all prior examples to train a new embedding in each learning session. *Finetune* represents the lower bound because it fine-tunes the embedding with examples only from the current learning session and has no means to prevent catastrophic forgetting.

Benchmark evaluations: Figure 2a summarizes the results of experiments using benchmark knowledge graph datasets of Table I (WN18RR, FB15K237), where the range of each measure is $[0, 1]$ and larger values are better. Although DGR significantly outperforms other methods in terms of inference (i.e., using ACC and FWT), there are insights and trade-offs to consider based on other factors.

- Figure 2b shows that DGR has a significantly lower learning speed (based on LCA) than the other methods since a new generative model must be trained in each learning session. If the number of epochs to train the generative model are ignored, DGR's LCA is comparable to Batch (DGR′ in Figure 2b) but still significantly lower than the regularization techniques (L2R and SI).



(a)

(b)

(c)

Fig. 2: Measures averaged for all datasets in Table I and graph embedding representations in Section IV. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color.

- Figure 2c indicates that methods with good inference performance also tend to have higher model memory growth (i.e., MS measure); among the methods with significantly better inference performance than Finetune, L2R has the smallest MS followed by SI and DGR.
- Since they regularize prior embeddings, L2R and SI initially perform better than DGR, as does the Batch baseline, as seen in the Hits@10 plots for each method at the start and end of each learning session (Figure 3).
- Figures 2 and 3 indicate that the CKGE methods based on architecture modification (i.e., PNN and CWR) have significantly lower inference performance than Finetune in all experiments. The difference in performance between PNN and the regularization-based methods shows the importance of flexibility over prior concepts for CKGE. Also, CWR's poor inference performance highlights the challenges of directly manipulating the embedding space because, although CWR can learn TE well in isolation, CE is quickly corrupted by the averaging performed to merge embeddings.

Service robotics evaluation: We constructed three evaluation scenarios using the AI2Thor knowledge graph dataset in Table II. Each scenario corresponds to a different class of semantics-driven robotics applications. The first scenario, *Unconstrained* in Figure 4a, corresponds to a robot that has access to all prior training examples at training time. More generally, this could represent robots with ready access to cloud services for data storage and processing. However, such a scenario may be unfeasible in some applications due to hardware constraints or security concerns. Our second scenario, *Data Constrained* in Figure 4b, represents robots

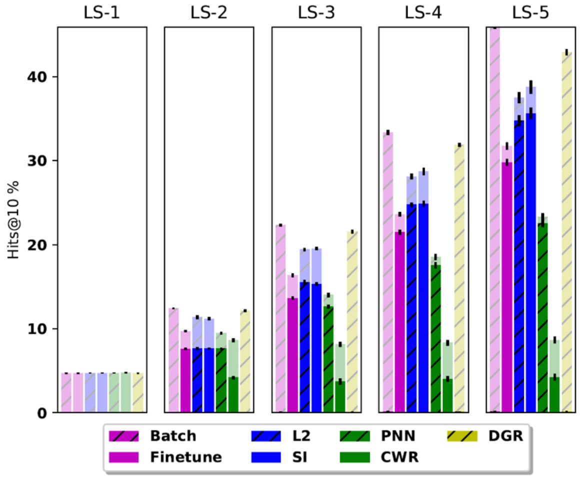
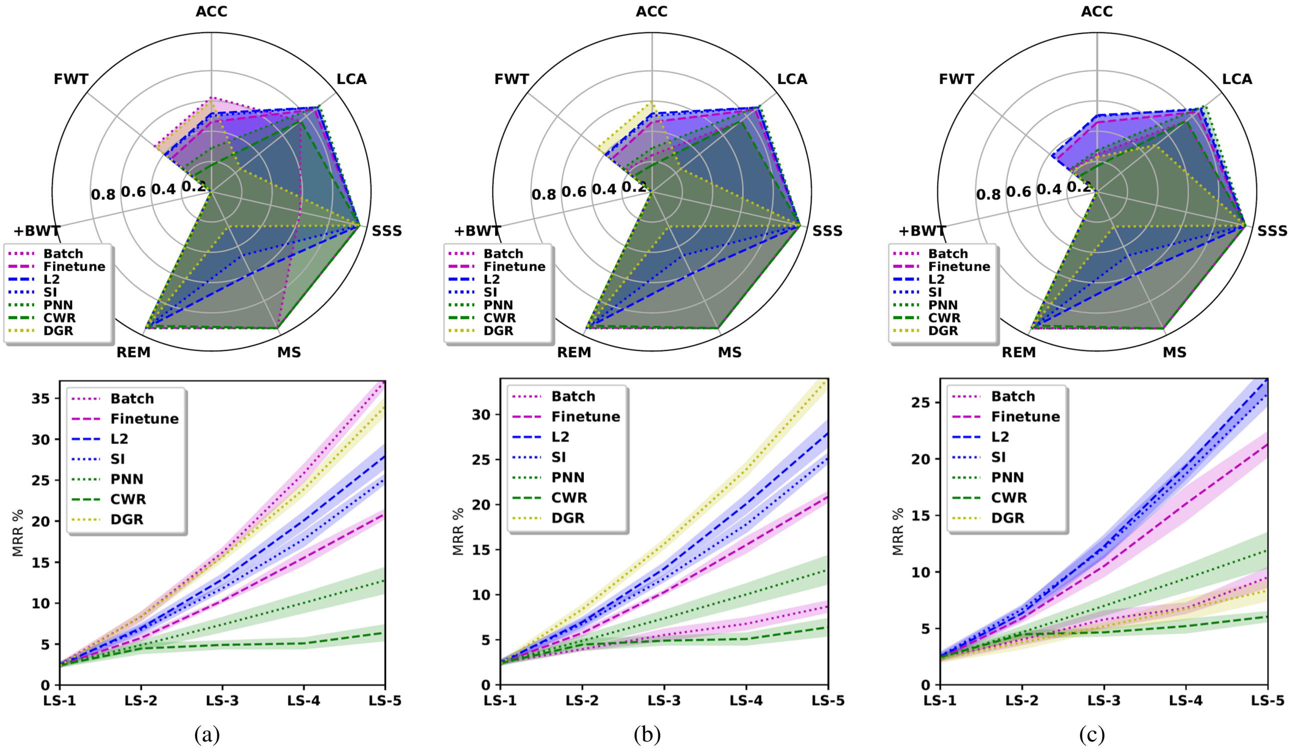


Fig. 3: Hits@10 from initial (bright) to final (transparent) epoch. Black errors bars indicate standard deviation. L2R and SI perform better than DGR in the initial epoch, but DGR outperforms in the final epoch after the first learning session. Best viewed in color.



(a) (b) (c)

Fig. 4: Semantics-driven robotics application scenarios: (a) Unconstrained; (b) Data Constrained; and (c) Time and Data Constrained. In each line plot, shading indicates standard deviation. Best viewed in color.

with access to limited training examples, e.g., only from the current learning session ($\mathcal{D}^{n}_{Tr}$); this could be due to storage constraints or dynamic domain changes. The final scenario, *Time and Data Constrained* in Figure 4c, mimics a mobile robot (or drone) operating under resource constraints; the robot only has access to training examples for the current learning session and has limited time to update the knowledge graph embedding. For simplicity, we limited the number of training epochs in each learning session to 100. The ranges of each measure are in $[0, 1]$ and larger values are better. The results from each scenario provide key insights about the choice of the CKGE method:

- In an *unconstrained scenario* (Figure 4a), such as one in which a robot might have access to a cloud compute service, Batch learning is the best choice despite its significantly lower sample efficiency (SSS) and learning speed (LCA) because it provides significantly higher ACC and FWT compared with other methods.
- In a *data constrained scenario* (Figure 4b), e.g., the robot can only update its semantic representation intermittently using limited on-board hardware. Batch's inference performance collapses because prior observations are unavailable. Given these constraints, DGR is the best choice, with much better ACC and FWT than other methods because it approximates Batch in the unconstrained scenario. However, DGR incurs a significant computational cost to train the generative model, resulting in a significantly lower LCA value.
- In a *data and time constrained scenario* (Figure 4c), e.g., the robot is updating its own semantic model on-board *during* a task, DGR is a poor choice because there is not enough time to sufficiently train the generative model. L2R and SI are better choices; SI with Analogy and L2R with either graph embedding offer significantly better inference performance than Finetune and significantly better LCA than Batch. Compared with SI, L2R's memory growth (MS) is significantly lower.

VI. Conclusion

Knowledge graph embeddings are increasingly being used as semantic representations in robotics applications, but it is difficult to update these representations incrementally. This paper introduced five representative continual learning-inspired methods for continual knowledge graph embedding (CKGE). We also introduced a heuristic sampling strategy and generated CKGE datasets based on benchmark knowledge graphs and a knowledge graph for the service robotics domain. Furthermore, we identified and built on measures for evaluating continual learning in robotics. We evaluated our embedding-generic methods on two knowledge graph embedding representations. Experimental evaluation using the benchmark knowledge graphs provided key insights characterizing the use of our CKGE methods in terms of factors such as inference, learning speed, and memory requirements. Our evaluation using the service robotics domain knowledge characterized the use of CKGE methods in three different classes of semantics-driven robotics applications. Future work will further investigate the adaptation of continual learning principles for CKGE in robot tasks that require semantic knowledge representations, including data from physical robots in complex, dynamic domains.

References

[1] N. Fulda, N. Tibbetts, Z. Brown, and D. Wingate, "Harvesting common-sense navigational knowledge for robotics from uncurated text corpora," in *Conference on Robot Learning*, 2017, pp. 525–534.

[2] J. Thomason, J. Sinapov, R. J. Mooney, and P. Stone, "Guiding exploratory behaviors for multi-modal grounding of linguistic descriptions," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] A. Daruna, W. Liu, Z. Kira, and S. Chetnova, "Robocse: Robot common sense embedding," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9777–9783.

[4] R. Scalise, J. Thomason, Y. Bisk, and S. Srinivasa, "Improving robot success detection using static object data," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4229–4235.

[5] D. Paulius, N. Eales, and Y. Sun, "A motion taxonomy for manipulation embedding," *Robotics: Science and Systems 2020*, 2020.

[6] J. Arkin, D. Park, S. Roy, M. R. Walter, N. Roy, T. M. Howard, and R. Paul, "Multimodal estimation and communication of latent semantic knowledge for robust execution of robot instructions," *The International Journal of Robotics Research*, p. 0278364920917755, 2020.

[7] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[8] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," *Proceedings of machine learning research*, vol. 70, p. 3987, 2017.

[9] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[10] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[11] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Conference on Robot Learning*, 2017, pp. 17–26.

[12] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[13] G. M. van de Ven and A. S. Tolias, "Generative replay with feedback connections as a general strategy for continual learning," *arXiv preprint arXiv:1809.10635*, 2018.

[14] H.-J. Song and S.-B. Park, "Enriching translation-based knowledge graph embeddings through continual learning," *IEEE Access*, vol. 6, pp. 60 489–60 497, 2018.

[15] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, "Don't forget, there is more than forgetting: new metrics for continual learning," *arXiv preprint arXiv:1810.13166*, 2018.

[16] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," *Proceedings of machine learning research*, vol. 70, p. 3987, 2017.

[17] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information Fusion*, vol. 58, pp. 52–68, 2020.

[18] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[19] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *International Conference on Machine Learning*, 2017, pp. 2168–2178.

[20] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoğlu, and G. Bartels, "Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 512–519.

[22] S. Chernova, V. Chu, A. Daruna, H. Garrison, M. Hahn, P. Khante, W. Liu, and A. Thomaz, "Situated bayesian reasoning framework for robots operating in diverse everyday environments," *International Foundation of Robotics Research*.

[23] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula, "Robobrain: Large-scale knowledge engine for robots," *arXiv preprint arXiv:1412.0691*, 2014.

[24] Y. Zhu, A. Fathi, and L. Fei-Fei, "Reasoning about object affordances in a knowledge base representation," in *European conference on computer vision*. Springer, 2014, pp. 408–424.

[25] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[26] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

[27] Y.-C. Hsu, Y.-C. Liu, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *arXiv preprint arXiv:1810.12488*, 2018.

[28] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey." *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[29] T. Wu, A. Khan, H. Gao, and C. Li, "Efficiently embedding dynamic knowledge graphs," *arXiv preprint arXiv:1910.06708*, 2019.

[30] P. Wang, S. Li, and R. Pan, "Incorporating gan for negative sampling in knowledge representation learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] L. Cai and W. Y. Wang, "Kbgan: Adversarial learning for knowledge graph embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1470–1480.

[32] Y. Dai, S. Wang, X. Chen, C. Xu, and W. Guo, "Generative adversarial networks based on wasserstein distance for knowledge graph embeddings," *Knowledge-Based Systems*, vol. 190, p. 105165, 2020.

[33] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 10–21.

[34] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *International Conference on Learning Representations*, 2018.